JSP - SESSION TRACKING

HTTP is a "stateless" protocol which means each time a client retrieves a Web page, the client opens a separate connection to the Web server and the server automatically does not keep any record of previous client request.

Still there are following three ways to maintain session between web client and web server:

Cookies:

A webserver can assign a unique session ID as a cookie to each web client and for subsequent requests from the client they can be recognized using the recieved cookie.

This may not be an effective way because many time browser does nots upport a cookie, so I would not recommend to use this procedure to maintain the sessions.

Hidden Form Fields:

A web server can send a hidden HTML form field along with a unique session ID as follows:

```
<input type="hidden" name="sessionid" value="12345">
```

This entry means that, when the form is submitted, the specified name and value are automatically included in the GET or POST data. Each time when web browser sends request back, then session_id value can be used to keep the track of different web browsers.

This could be an effective way of keeping track of the session but clicking on a regular (<A HREF...>) hypertext link does not result in a form submission, so hidden form fields also cannot support general session tracking.

URL Rewriting:

You can append some extra data on the end of each URL that identifies the session, and the server can associate that session identifier with data it has stored about that session.

For example, with http://tutorialspoint.com/file.htm;sessionid=12345, the session identifier is attached as sessionid=12345 which can be accessed at the web server to identify the client.

URL rewriting is a better way to maintain sessions and works for the browsers when they don't support cookies but here drawback is that you would have generate every URL dynamically to assign a session ID though page is simple static HTML page.

The session Object:

Apart from the above mentioned three ways, JSP makes use of servlet provided HttpSession Interface which provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.

By default, JSPs have session tracking enabled and a new HttpSession object is instantiated for each new client automatically. Disabling session tracking requires explicitly turning it off by setting the page directive session attribute to false as follows:

```
<%@ page session="false" %>
```

The JSP engine exposes the HttpSession object to the JSP author through the implicit **session** object. Since **session** object is already provided to the JSP programmer, the programmer can immediately begin storing and retrieving data

from the object without any initialization or getSession().

Here is a summary of important methods available through session object:

S.N.	Method & Description
1	**public Object getAttribute(String name)** This method returns the object bound with the specified name in this session, or null if no object is bound under the name.
2	**public Enumeration getAttributeNames()** This method returns an Enumeration of String objects containing the names of all the objects bound to this session.
3	**public long getCreationTime()** This method returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.
4	**public String getId()** This method returns a string containing the unique identifier assigned to this session.
5	**public long getLastAccessedTime()** This method returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT.
6	**public int getMaxInactiveInterval()** This method returns the maximum time interval, in seconds, that the servlet container will keep this session open between client accesses.
7	**public void invalidate()** This method invalidates this session and unbinds any objects bound to it.
8	**public boolean isNew(** This method returns true if the client does not yet know about the session or if the client chooses not to join the session.
9	**public void removeAttribute(String name)** This method removes the object bound with the specified name from this session.
10	**public void setAttribute(String name, Object value)** This method binds an object to this session, using the name specified.
11	**public void setMaxInactiveInterval(int interval)** This method specifies the time, in seconds, between client requests before the servlet container will invalidate this session.

Session Tracking Example:

This example describes how to use the HttpSession object to find out the creation time and the last-accessed time for a session. We would associate a new session with the request if one does not already exist.

```
<%@ page import="java.io.*,java.util.*" %>
<%
   // Get session creation time.
```

```
   Date createTime = new Date(session.getCreationTime());
   // Get last access time of this web page.
   Date lastAccessTime = new Date(session.getLastAccessedTime());

   String title = "Welcome Back to my website";
   Integer visitCount = new Integer(0);
   String visitCountKey = new String("visitCount");
   String userIDKey = new String("userID");
   String userID = new String("ABCD");

   // Check if this is new comer on your web page.
   if (session.isNew()){
      title = "Welcome to my website";
      session.setAttribute(userIDKey, userID);
      session.setAttribute(visitCountKey,  visitCount);
   }
   visitCount = (Integer)session.getAttribute(visitCountKey;
   visitCount = visitCount + 1;
   userID = (String)session.getAttribute(userIDKey);
   session.setAttribute(visitCountKey,  visitCount);
%>
<html>
<head>
<title>Session Tracking</title>
</head>
<body>
<center>
<h1>Session Tracking</h1>
</center>
<table border="1" align="center">
<tr bgcolor="#949494">
   <th>Session info</th>
   <th>Value</th>
</tr>
<tr>
   <td>id</td>
   <td><% out.print( session.getId()); %></td>
</tr>
<tr>
   <td>Creation Time</td>
   <td><% out.print(createTime); %></td>
</tr>
<tr>
   <td>Time of Last Access</td>
   <td><% out.print(lastAccessTime); %></td>
</tr>
<tr>
   <td>User ID</td>
   <td><% out.print(userID); %></td>
</tr>
<tr>
   <td>Number of visits</td>
   <td><% out.print(visitCount); %></td>
</tr>
</table>
</body>
</html>
```

Now put above code in main.jsp and try to access *http://localhost:8080/main.jsp*. It would display the following result when you would run for the first time:

WELCOME TO MY WEBSITE

Session Infomation

Session info	value

id	0AE3EC93FF44E3C525B4351B77ABB2D5
Creation Time	Tue Jun 08 17:26:40 GMT+04:00 2010
Time of Last Access	Tue Jun 08 17:26:40 GMT+04:00 2010
User ID	ABCD
Number of visits	0

Now try to run the same JSP for second time, it would display following result.

WELCOME BACK TO MY WEBSITE

Session Infomation

info type	value
id	0AE3EC93FF44E3C525B4351B77ABB2D5
Creation Time	Tue Jun 08 17:26:40 GMT+04:00 2010
Time of Last Access	Tue Jun 08 17:26:40 GMT+04:00 2010
User ID	ABCD
Number of visits	1

Deleting Session Data:

When you are done with a user's session data, you have several options:

- **Remove a particular attribute:** You can call *public void removeAttribute(String name)* method to delete the value associated with a particular key.
- **Delete the whole session:** You can call *public void invalidate()* method to discard an entire session.
- **Setting Session timeout:** You can call *public void setMaxInactiveInterval(int interval)* method to set the timeout for a session individually.
- **Log the user out:** The servers that support servlets 2.4, you can call **logout** to log the client out of the Web server and invalidate all sessions belonging to all the users.
- **web.xml Configuration:** If you are using Tomcat, apart from the above mentioned methods, you can configure session time out in web.xml file as follows.

```
<session-config>
  <session-timeout>15</session-timeout>
</session-config>
```

The timeout is expressed as minutes, and overrides the default timeout which is 30 minutes in Tomcat.

The getMaxInactiveInterval( ) method in a servlet returns the timeout period for that session in seconds. So if your session is configured in web.xml for 15 minutes, getMaxInactiveInterval( ) returns 900.